HAL
open science

THE RISE OF THE LOTTERY HEROES: WHY ZERO-SHOT PRUNING IS HARD

Enzo Tartaglione

To cite this version:

Enzo Tartaglione. THE RISE OF THE LOTTERY HEROES: WHY ZERO-SHOT PRUNING IS HARD. IEEE International Conference on Image Processing (ICIP 22), Oct 2022, Bordeaux, France. hal-03766740

HAL Id: hal-03766740

https://hal.science/hal-03766740v1

Submitted on 1 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THE RISE OF THE LOTTERY HEROES: WHY ZERO-SHOT PRUNING IS HARD

Enzo Tartaglione

LTCI, Télécom Paris, Institut Polytechnique de Paris

ABSTRACT

Recent advances in deep learning optimization showed that just a subset of parameters are really necessary to successfully train a model. Potentially, such a discovery has broad impact from the theory to application; however, it is known that finding these trainable sub-network is a typically costly process. This inhibits practical applications: can the learned sub-graph structures in deep learning models be found at training time? In this work we explore such a possibility, observing and motivating why common approaches typically fail in the extreme scenarios of interest, and proposing an approach which potentially enables training with reduced computational effort. The experiments on either challenging architectures and datasets suggest the algorithmic accessibility over such a computational gain, and in particular a trade-off between accuracy achieved and training complexity deployed emerges.

***Index Terms*—** The lottery ticket hypothesis, pruning, computational complexity, deep learning

1. THE ELEPHANT IN THE ROOM

Artificial neural networks (ANNs) are nowadays one of the most studied algorithms used to solve a huge variety of tasks. Their success comes from their ability to learn from examples, not requiring any specific expertise and using very general learning strategies. However, deep models share a common obstacle: the large number of parameters, which allows their successful training [1, 2], determines high training costs in terms of computation. For example, a ResNet-18 trained on ILSVRC'12 with a standard learning policy [3], requires operations in the orders of hundreds of PFLOPs for back-propagation, or even efficient architectures like MobileNet-v3 [4] on smaller datasets like CIFAR-10 with an efficient learning policy [5], require order of hundreds of TFLOPs for back-propagation! Despite an increasingly broad availability of powerful hardware to deploy training, energetic end efficiency issues still need to be addressed.

Accepted for publication at the IEEE International Conference on Image Processing (IEEE ICIP 2022).

© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

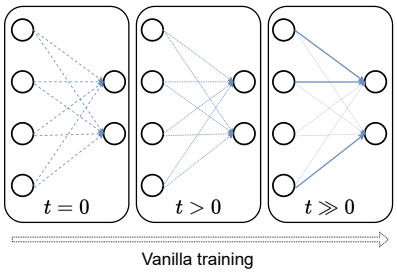
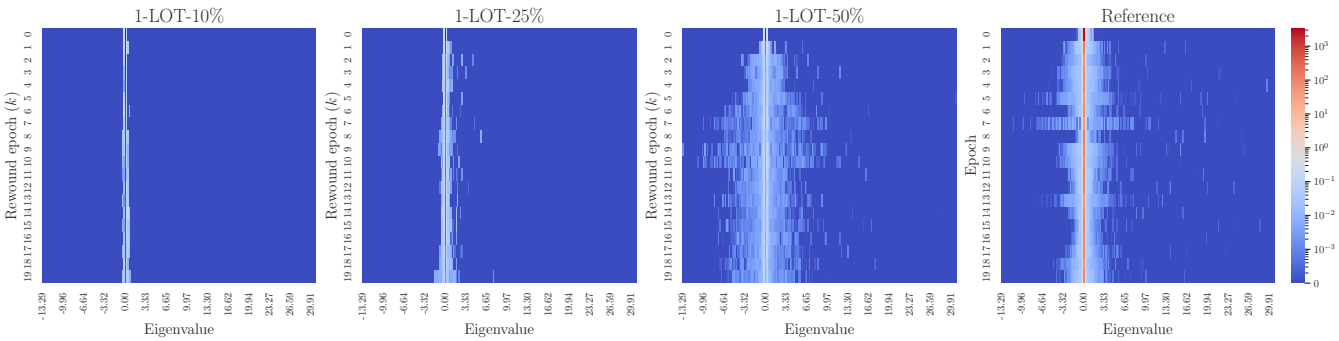


Fig. 1: A subset of parameters, sufficient to reach good generalization, is typically determined in an iterative fashion. Can they be determined earlier, during a normal vanilla training?

Some approaches have been proposed in order to reduce the computational complexity for deep neural networks. The act of removing parameters (or entire units) from a deep neural network is named *pruning*. Despite the first works have been proposed many decades ago [6], pruning became popular just a few years ago, targeting the reduction of the model's size at deployment time and making inference more efficient [7, 8, 9, 10, 11].

A recent work, the *lottery ticket hypothesis* [12], suggests that the fate of a parameter, namely whether it is useful for training (winner at the lottery of initialization) or if it can be removed from the architecture, is decided already at the initialization step. Frankle and Carbin propose experiments showing that, with an a-posteriori knowledge of the training over the full model, it is possible to identify these parameters, and that it is possible to successfully perform a full training just with them, matching the performance of the full model. However, in order to identify these winners, a costly iterative pruning strategy is deployed, meaning that the complexity of finding the lottery winners is larger than training the full model. Is it possible to deploy a zero-shot strategy, where we identify the lottery winners before, or during, the training of the model itself, to get a real computational advantage?

In this work we ground the lottery ticket hypothesis, motivating why the originally proposed strategy, despite showing the existence of the lottery tickets, is computationally sub-optimal. We leverage over experiments on CIFAR-10 and ILSVRC'12, qualitatively and quantitatively, analyzing the



(a) (b) (c) (d)

Fig. 2: Example of distribution of the eigenvalues of the Hessian matrix calculated on the CIFAR-10 training set for ResNet-32 along different rewound epochs (k) retaining the 10% of the parameters (a), the 25% (b) the 50% (c) and all the parameters (d). Here $I = 1$. The represented scenario is qualitatively matched for different initialization of the model.

loss landscape evolution and proposing a strategy which opens the road to the design of optimization strategies which can effectively save computational power at training time. The lottery tickets are not evident in the first epochs, but they rise when the model's parameters have reached a specific subspace, and that iterative pruning strategies, which are necessary for traditional lottery ticket approaches, are not necessary to identify the lottery winners (Fig. 1). We observe the feasibility of having a pruning strategy on-going at training time, and that, in very high compression regimes, the performance is mainly bound by the computational complexity budget we are willing to deploy.

2. THE LOTTERY OF THE INITIALIZATION

The lottery ticket hypothesis. It is a known fact that deep neural networks are typically over-parametrized and, after training, a part of the parameters can be removed without harming the performance, or even slightly improving the performance in small pruning regimes [7, 13]. However, an interesting question rises: is it possible to train a subnetwork, still achieving the same performance as training the full model? In their famous paper, Frankle and Carbin provide a method to identify, from a the set of initialized parameters, a subset $\mathcal{W}$ of the parameters which are sufficient, when trained in isolation, to achieve the same performance of the full model [12]. From a practical perspective, this means that all the parameters not in $\mathcal{W}$ (hence in $\overline{\mathcal{W}}$) are pruned from the model, or more concretely, their value is locked to zero. Let us say that a trained model has N parameters: we define a mask $\mathcal{M} \in \{0;1\}^N$ of the same dimensionality of the original model's parameters $W \in \mathbb{R}^N$ and we say that a parameter $w_i \in \mathcal{W} \Leftrightarrow m_i = 1 | m_i \in \mathcal{M}$.[1] Alg. 1 reports the algorithm to find the lottery winners, which involves an iterative magnitude pruning strategy (IMP, line 7): after every training round (line 6) the lowest $(100-R)\% \in \mathcal{W}$ having the smallest magnitude will be removed from $\mathcal{W}$. The parameters in $\mathcal{W}$ will then be *rewound* to their original values (line 5) and a new training, just updating $\mathcal{W}$, will be performed:

$$w_i^{t+1} = \begin{cases} w_i^t - u_i^t & if\ w_i \in \mathcal{W} \\ 0 & if\ w_i \in \overline{\mathcal{W}}, \end{cases} \quad (1)$$

where u_i is some generic update term. In principle, the parameters in $\overline{\mathcal{W}}$ are not in the model, and for instance they should not be included in the computation anymore; however, we still need to encode that are missing, producing an overhead, as they are removed in an unstructured way [9].[2]

Limits. Despite achieving the purpose of showing that winning tickets exist, there is a major, significant drawback of the approach in Alg. 1: the complexity of the overall strategy, namely the number of rewinds I to converge to the target minimal subset $\mathcal{W}$, which depends on the amount of remaining parameters R. Such a value can not be set to very high

Algorithm 1 Lottery winners in I iterations with $R\%$ remaining parameters at every iteration (I-LOT-R).

1: **procedure** I-LOT-R(W^0, R, I)
2: $i \leftarrow 0$
3: $\mathcal{M} \leftarrow 1$ ▷ unit vector
4: **while** $i < I$ **do**
5: $W^0_{LOT} \leftarrow W^0 \cdot \mathcal{M}$
6: $W^f_{LOT} \leftarrow$ TRAIN(W^0_{LOT}, $\mathcal{M}$) ▷ (1)
7: $\mathcal{M} \leftarrow$ MAGNITUDE PRUNE(W^f_{LOT}, R, $\mathcal{M}$)
8: $i \leftarrow i + 1$
9: **end while**
10: **return** $\mathcal{M}$
11: **end procedure**

[1] we index the parameters of the models along a unique vector for simplicity.

[2] unless entire structures are not entirely removed from the model, but this is not the general case.

Algorithm 2 Lottery winners with k epochs warm-up.

procedure I-LOT-R WITH WARM-UP(W^0, R, I, k)
  $e \leftarrow 0$
  $W^e \leftarrow W^0$
  while $e < k$ **do**
    $W^e \leftarrow$ TRAIN ONE EPOCH(W^e) ▷ Here all the weights are trained, for one epoch only
    $e \leftarrow e + 1$
  end while
  $W^k \leftarrow W^e$
  $\mathcal{M} \leftarrow$ I-LOT-R(W^k, R, I)
  return $\mathcal{M}$
end procedure

Algorithm 3 Rise of the lottery heroes with $R\%$ remaining parameters (RISE-R).

procedure RISE-R(W^k, R)
  $W^f \leftarrow$ TRAIN(W^k)
  $\mathcal{M} \leftarrow$ MAGNITUDE PRUNE(W^f, R)
  $W^f_{RISE} \leftarrow$ TRAIN(W^k, $\mathcal{M}$) ▷ (2)
  return $\mathcal{M}$
end procedure

values, as the approach fails. In order to improve this aspect, more works have tried to address possible solutions. In particular, [14] shows that there is a region, at the very early stages of learning, where the lottery tickets identified with iterative pruning are not stable (if they are found, for different seeds they are essentially different). The novelty here introduced is an inspection over the epoch (or mini-batch iteration) where to rewind: simply, we pass to Alg. 1 the parameters of a model already trained for the first k epochs (Alg. 2). This is endorsed also by other works, like [15, 16, 17, 18], while other works reduce the overall complexity of the iterative training by drawing early-bird tickets [19] (meaning that they learn the lottery tickets when the model have not yet reached full convergence) or even reducing the training data [20].

Preliminary experiment and analysis. The golden mine in this context would be to address a strategy for zero-shot lottery drafting, meaning that the lottery tickets are identified before the training itself. In order to assess its feasibility, let us define a companion model (ResNet-32) trained on CIFAR-10 for 180 epochs, using SGD optimization with initial learning rate 0.1 and decayed by a factor 0.1 at milestones 80 and 120, with momentum 0.9, batch size 100 and weight decay $5 \cdot 10^{-5}$, as in [21]. Fig. 2 reports the distribution of the eigenvalues of the Hessian computed for the first 19 epochs on $W^k_{LOT} = W^k \cdot \mathcal{M}$ (Alg. 2) with $I = 1$, evaluated on the full training set. We are interested in this specific one-shot scenario as we explore the *possibility* of removing trained parameters *during training* towards computational complexity saving, and in the one-shot scenario we remove them from

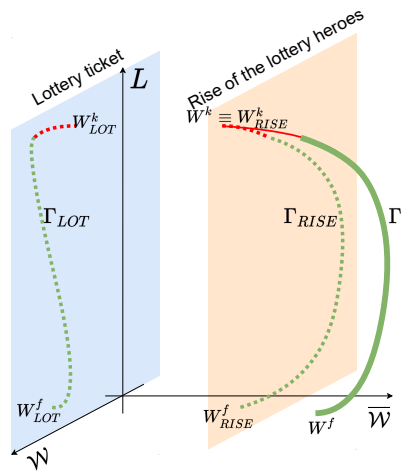


Fig. 3: LOT projects the parameters in the subspace $\mathcal{W}$: for low R the learning trajectory Γ_{LOT} is very steep making the optimization problem hard, compared to the trajectory Γ of the full model. RISE, on the contrary, does not project the parameters, but constrains the optimization problem to the parameters identified by $\mathcal{M}$.

the original, vanilla training trajectory. For this experiment, the `pyhessian` library has been used [22], along with a NVIDIA A40 GPU. We observe that, compared to the reference (namely, the distribution of the eigenvalues evaluated on the full model - Fig. 2d) when R is low ($R = 10\%$ - Fig. 2a - or $R = 25\%$ - Fig. 2b), the distribution changes significantly. In particular, a peak to values close to zero is observed: locally, the loss landscape is flat. Contrarily, for a higher R regime (Fig. 2c) the distribution is richer and similar to the reference (Fig. 2d). When the loss landscape becomes flatter, the optimization problem itself is harder. We observe indeed that, with respect to a baseline performance of 92.92% on the test set, with $R = 10\%$, despite rewinding up to $k = 20$, the achieved performance is never above 60%. Why does this happen? In the next section we tackle this problem motivating why it is hard to evaluate the winning tickets when $I = 1$ (or simply, in a one-shot fashion).

3. WINNING TICKETS IN HINDSIGHT: THE RISE OF THE LOTTERY HEROES

The rise of the lottery heroes. Fig. 3 portraits the learning optimization constraint when pruning at initialization. When sampling the tickets and then rewinding, the model itself does not preserve the same initialization W^k, but it will be re-initialized a projection W^k_{LOT}, and its optimization is enforced in the subspace $\mathcal{W}$ (light blue). Despite such an approach does not introduce big problems in high R regimes,[3]

[3] because the introduced perturbation ΔW^k in the initialization is small or, when I grows, such perturbation is beneficially polarized towards removing parameters valuing approximately zero at the end of the training.

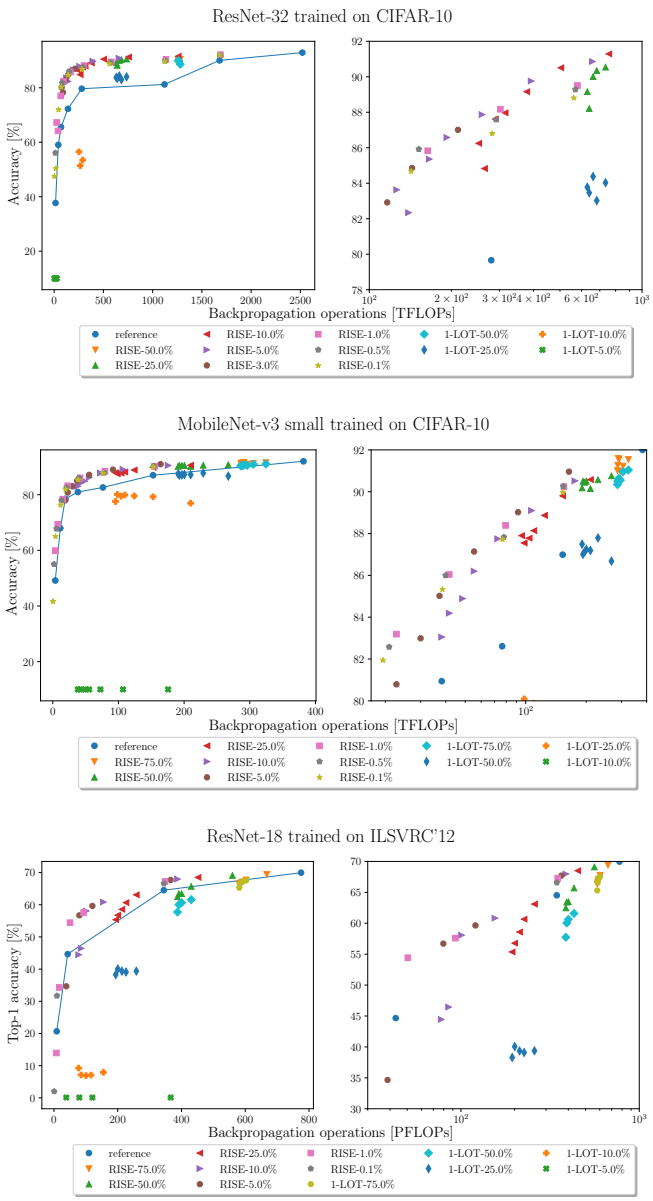


Fig. 4: Training results for ResNet-32 trained on CIFAR-10 (top), MobileNet-v3 small on CIFAR-10 (center) and ResNet-18 on ILSVRC'12 (bottom).

in low R regimes the optimization problem is harder: the loss landscape becomes locally flat (Fig. 2) and the optimization problem can not be easily solved. However, we can "lock" the non-winning parameters and let the potential winners to rise and to evolve towards their final value, constraining the optimization problem for the values determined by $\mathcal{M}$ and freezing the others (light orange). Towards this end, we can modify the update rule in (1) to

$$w_i^{t+1} = \begin{cases} w_i^t - u_i^t & if\ w_i \in \mathcal{W} \\ w_i^k & if\ w_i \in \overline{\mathcal{W}}. \end{cases} \quad (2)$$

Using this approach, we will no longer incur in the same obstacles as in Sec. 2, as we will optimize starting from the exact same loss landscape (Alg. 3).

Experiments. In order to validate our approach, we run the following experiments: i) ResNet-32 trained on CIFAR-10 with same setup as described in Sec. 2; ii)MobileNet-v3 small in CIFAR-10 with training for 100 epochs with 5 epochs linear warm-up followed by cosine annealing (from learning rate 0.35), optimized with SGD with momentum 0.9 weight decay 6e-5 and batch size 128, learning rate tuning as in [5]; iii) ResNet-18 on ILSVRC'12 with training for 90 epochs with initial learning rate 0.1 and decayed by a factor 0.1 at milestones 30 and 60, optimized with SGD with momentum 0.9 batch size 1024 and weight decay $5 \cdot 10^{-5}$, same setup as in [3]. All the results are reported in Fig. 4. On the left the full results are displayed, on the right a zooming on the mostly dense regions is proposed, in log-scale. The continuous blue line is the reference training with the full model. Back-propagation operations are evaluated on the training complexity for one complete training. Every point in every graph represents a complete full training: the final performance achieved is reported. The multiple points with same color/shape refer to different k value (refer to Alg. 2 - for RISE line 9 calls RISE-R): as k increases, the back-propagation operations increase, as more training on the full model is required.

Unsurprisingly, we observe low performance for 1-LOT with low R, and despite different values of rewind, for low R values the performance is heavily sub-optimal (like for $R = 25\%$ in ResNet-32/CIFAR-10). On the contrary, even with extremely low R regimes, we observe a progressive increment in the performance as k increases. Notably, in the accuracy-backpropagation complexity plane, a Pareto-like curve is drawn by RISE: what emerges is that not the rewound epoch k, nor R are really the metrics to determine the final performance of the model, but the training complexity deployed itself. Indeed, for low training complexity RISE achieves similar performance regardless of R or k, under similar back-propagation complexity.

4. FORECASTING THE RISE OF THE LOTTERY HEROES?

In this work we have observed that traditional lottery ticket approaches are likely to fail in extreme scenarios when just a small subset of parameters is trained. However, locking the "non-winning" parameters and allowing the winners to evolve in the original loss landscape is a winning strategy. With such an approach it is possible to target a desired training performance training just a minimal portion of the entire model. In particular, the governing metrics in extreme regimes is the deployed training complexity. The results presented in this work, validated on standard architectures (ResNet), on already compact architectures trained with complex policies

(MobileNet-v3) and on state-of-the-art datasets (ILSVRC'12) open the research towards the possibility of effectively deploying heavy computational saving at training time, as just a few directions are needed to train the model: the directions where the lottery heroes rise. Next work includes the identification of these directions at training time, as this work showed these exist and are algorithmically accessible.

5. REFERENCES

[1] Jimmy Ba and Rich Caruana, "Do deep nets really need to be deep?," in *Advances in neural information processing systems*, 2014, pp. 2654–2662.

[2] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, 2014, pp. 1269–1277.

[3] "Imagenet training in pytorch," `https://github.com/pytorch/examples/tree/master/imagenet`, 2022.

[4] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al., "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.

[5] "Mobilenetv3 - an implementation of mobilenetv3 with pytorch," `https://github.com/ShowLo/MobileNetV3`, 2020.

[6] Yann LeCun, John S Denker, and Sara A Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[7] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[8] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *34th International Conference on Machine Learning, ICML 2017*, 2017, vol. 5, pp. 3854–3863, cited By 29.

[9] Andrea Bragagnolo, Enzo Tartaglione, Attilio Fiandrotti, and Marco Grangetto, "On the role of structured pruning for neural network compression," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3527–3531.

[10] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[11] Enzo Tartaglione, Andrea Bragagnolo, Francesco Odierna, Attilio Fiandrotti, and Marco Grangetto, "Serene: Sensitivity-based regularization of neurons for structured sparsity in neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[12] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[13] Enzo Tartaglione, Andrea Bragagnolo, and Marco Grangetto, "Pruning artificial neural networks: A way to find well-generalizing, high-entropy sharp minima," in *International Conference on Artificial Neural Networks*. Springer, 2020, pp. 67–78.

[14] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin, "Linear mode connectivity and the lottery ticket hypothesis," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3259–3269.

[15] Jonathan Frankle, David J Schwab, and Ari S Morcos, "The early phase of neural network training," *arXiv preprint arXiv:2002.10365*, 2020.

[16] Ari Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian, "One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers," *Advances in neural information processing systems*, vol. 32, 2019.

[17] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir, "Proving the lottery ticket hypothesis: Pruning is all you need," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6682–6691.

[18] Sharath Girish, Shishira R Maiya, Kamal Gupta, Hao Chen, Larry S Davis, and Abhinav Shrivastava, "The lottery ticket hypothesis for object recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 762–771.

[19] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G Baraniuk, Zhangyang Wang, and Yingyan Lin, "Drawing early-bird tickets: Towards more efficient training of deep networks," *arXiv preprint arXiv:1909.11957*, 2019.

[20] Zhenyu Zhang, Xuxi Chen, Tianlong Chen, and Zhangyang Wang, "Efficient lottery ticket finding: Less data is more," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12380–12390.

[21] "Proper resnet implementation for cifar10/cifar100 in pytorch," https://github.com/akamaster/pytorch_resnet_cifar10, 2020.

[22] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney, "Pyhessian: Neural networks through the lens of the hessian," in *2020 IEEE international conference on big data (Big data)*. IEEE, 2020, pp. 581–590.